Amendments to the Specification:

Please replace paragraphs [0008]-[0009] with the following replacement paragraphs.

[0008] On-chip debugging hardware 12 recognizes certain debug commands that are received onto the on-chip debugging hardware 12 in serial fashion through a serial interface of the on-chip debugger. One of the commands is a command to write to a program memory location. Debug routine 9 therefore causes hardware debug device 1 to send the appropriate write_to_program_memory commands to the on-chip debugging hardware 12 such that the on-chip debugging hardware causes a block of code ~~is~~**to be** written into the appropriate block of program memory, one address location at a time. The commands understood by on-chip debugging hardware 12 of the target that cause the sub-actions to be carried out are referred to here as "microcommands." The commands sent from host computer 3 to hardware debug device 1 that instruct the hardware debug device to perform the higher level action are referred to here as "macrocommands."

[0009] Typical on-chip debugging hardware supports numerous microcommands other than the write_to_program_memory microcommand. Such microcommands may include, for example, microcommands to stop the target processor, single-step the target processor, read and write to various internal registers within the target processor, read and write to data memory, set breakpoints in code executed by the target processor, stuff instructions into the target

processor, read and write watchpoints, read status registers, and otherwise exercise and monitor the target processor. These microcommands are usable to carry out sub-actions involved in executing other macrocommands.

Please replace paragraph [0036] with the following replacement paragraph.

[0036] Figure 10 is a table that shows ~~how~~ the compression encoding used to send information from the hardware debug device to the host computer in accordance with one embodiment.

Please replace paragraph [0044] with the following replacement paragraph.

[0044] In one embodiment, a manufacturer of microcontrollers and/or microprocessors maintains a network accessible development resource 110 involving hardware debug device 101 and one or more target processors 106. In this case, the target processors may not be embedded into a new system, but rather are provided on ~~standare~~standard ~~developmenb~~development printed circuit boards. The network accessible development resource 110 is accessible via the internet through network 102. A user (such as a potential purchaser of microcontrollers) located in a remote location can then access the network accessible development resource 110 and work with a desired target processor without having to obtain or set up a hardware debug device and a target processor.

Please replace paragraphs [0052]-[0053] with the following replacement paragraphs.

[0052] The DTLs syntax supports the efficient reading and writing of registers in the target, the efficient reading and writing of large blocks of memory on the target, and auto-incrementing of addresses within the on-chip debugging logic of the target. The DTLs syntax also supports querying by the host of the DTLi interpreter to receive and an indication[[:]] of: the DTLs version that is supported, the maximum length of a DTLs data buffer, the maximum number of DTL variables, and the success or failure of each DTL script. DTLs and the DTLi interpreter support at least ten levels of nesting of parenthetical expressions and at least ten levels of nested loops.

[0053] All operations within the DTLs language are fully blocking such that no operating system support is needed or used. Accordingly, the DTLi interpreter is a monolithic block of statistically linked compiled C/C++ code that executes on the hardware debug device without any operating system.